Eléments d'architecture des systèmes complexes

Daniel Krob [1]

1. Introduction

1.1 Le défi de la complexité

La réalisation d'un grand système technique moderne nécessite la mise en place de processus d'ingénierie complexes allant parfois jusqu'à impliquer plusieurs milliers d'ingénieurs issus de nombreuses spécialités différentes. Dans l'automobile, un projet « véhicule » représente ainsi actuellement typiquement à une charge de travail de près de 1.500 hommes-années répartie sur 3 à 4 ans et fait intervenir 30 à 50 corps de métiers pour un coût global d'investissement de l'ordre du milliard d'euros. Les grands projets de déploiement de systèmes d'information ont des dimensionnements du même ordre de grandeur : la gestion de la fusion des systèmes d'information de deux grandes banques françaises a par exemple nécessité récemment plus de six mois d'études préliminaires et de deux ans de mise en oeuvre, en impliquant un millier d'ingénieurs informaticiens à temps complet et un budget d'environ 400 millions d'euros, pour faire converger près de 250 sous-systèmes informatiques différents. On pourrait bien entendu multiplier les autres exemples du même ordre car la réalité est partout la même : les grands systèmes technologiques modernes sont caractérisés par une énorme complexité technique intrinsèque qui ne semble malheureusement qu'augmenter au fur et à mesure du temps.

Comme on s'en doute, la conception et la mise en place de ces grands systèmes techniques n'est pas sans poser des difficultés conceptuelles et techniques majeures car il est de plus en plus difficile – voire parfois impossible – pour un individu d'appréhender totalement de tels systèmes dans leur globalité, d'où leur dénomination de *systèmes complexes*. La notion de *complexité d'un système industriel* est certes floue et subjective de ce point de vue, mais ce concept renvoie à une réalité extrêmement forte et pragmatique : il est en effet naturellement associé aux systèmes techniques dont la maîtrise de la conception, de l'industrialisation, de la maintenance et de l'évolution pose des problèmes importants et difficiles d'*intégration*, directement liés au grand nombre de leurs composants élémentaires (logiciels, matériels ou humains) et à leur importante hétérogénéité scientifique et technologique, ce qui les rend de moins en moins facilement maîtrisable dans leur ensemble par l'être humain.

1.2 L'ingénierie système

Pour faire face à cette complexité, il a bien sûr fallu construire des méthodologies adaptées. Leur émergence prend son origine dans l'après seconde guerre mondiale où les Etats Unis ont dû faire face aux contraintes de la guerre froide : celles-ci les ont en effet conduit à construire des systèmes de défense où la problématique de la gestion des informations, des prises de décision et des ripostes militaires avait été prise en compte dans sa globalité de manière cohérente et intégrée pour pouvoir garantir un délai de réaction rapide entre la détection d'une éventuelle menace soviétique et une contre-attaque américaine. Les nécessités opérationnelles ont donc naturellement obligé les ingénieurs américains à concevoir et à mettre en œuvre des

[1] CNRS & Ecole Polytechnique – LIX – 91128 Palaiseau Cedex – France – email : dk@lix.polytechnique.fr.
Cet article a été soutenu par la chaire Ecole Polytechnique – Thales « Ingénierie des systèmes complexes ».

réponses sur les plans techniques et managériaux appropriées aux nombreuses problématiques d'intégration des différents composants (en l'occurrence les systèmes de renseignement et d'observation, les centres de commandement, les systèmes de riposte, etc.) des systèmes de défense globaux qu'ils devaient mettre en place.

Un corpus de connaissances qui a pris très tôt la dénomination – toujours en vigueur – d'*ingénierie système*, centré sur la maîtrise de l'intégration des grands systèmes industriels, s'est ainsi progressivement formé dans ce contexte au cours des années 50, 60 et 70 qui ont notamment vues la publication d'un grand nombre d'ouvrages de référence, maintenant classiques, qui ont contribué à structurer ce domaine [2] . L'ingénierie système regroupe donc logiquement désormais l'ensemble des *concepts*, des *méthodologies* et des *bonnes pratiques organisationnelles et techniques* que le monde industriel a dû développer pour arriver à maîtriser la complexité de la conception et de la fabrication des systèmes technologiques modernes (voir [1], [16], [18], [19], [23] ou [26] pour plus de détails à ce sujet).

Il est notamment important de comprendre que l'ingénierie système joue sur deux registres principaux fortement couplés que cette discipline pense donc de manière intégrée , à savoir :

- une dimension *technique*, centrée sur le *système industriel* (en tant que tel) que l'on veut construire, qui est sous-tendue par une grille d'analyse systémique ;
- une dimension *organisationnelle et managériale* qui est centrée sur le *système de fabrication* du système industriel que l'on cherche à réaliser.

De nombreux autres processus techniques ou managériaux clefs (ingénierie des exigences, modélisation et dimensionnement systémique, ingénierie des interfaces, gestion logistique intégrée, management des équipes et des projets systèmes, gouvernance système, gestion de configuration, etc.), qui peuvent être analysés selon ces deux dimensions (cf. aussi Figure 1), sont bien entendu également du ressort de l'ingénierie système. Nous renvoyons notamment aux excellents ouvrages de référence déjà cités plus haut pour plus de détails sur ces sujets.

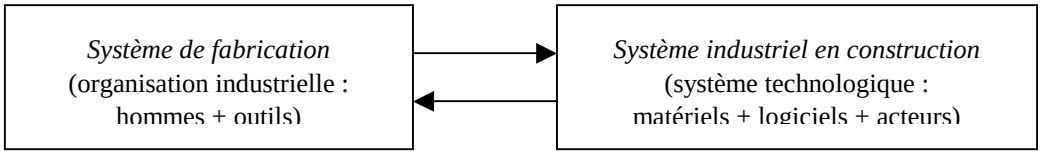

Figure 1 – Les deux systèmes structurant la démarche d'ingénierie système

1.3 L'approche système

L'ingénierie système n'est cependant, comme son nom l'indique d'ailleurs parfaitement, que l'application d'un paradigme de pensée – l'*approche système* – à l'ingénierie des systèmes industriels. Il n'est donc pas totalement inutile de présenter rapidement ce que nous attendons par « approche système », surtout dans la mesure où celle-ci est au cœur de notre travail [3] .

Comme nous venons tout juste de le dire, l'approche système est donc d'abord un paradigme de pensée, c'est-à-dire une manière d'appréhender de façon homogène une certaine catégorie d'objets « dynamiques » – les systèmes – que l'on cherche typiquement à étudier, à construire

[2] Voir le chapitre introductif de [12] pour en savoir plus sur l'histoire de l'ingénierie système.

[3] Attention : le terme « système » n'a pas du tout le même sens quand il s'agit d'un nom, auquel cas il désigne une catégorie d'objets – les systèmes – ou s'il est adjectivé, auquel cas cette terminologie signifie « vision de haut niveau d'un système », autrement dit fait référence à l'approche système.

ou à faire évoluer (selon le cas) en prenant systématiquement un *point de vue global*. Ce dernier élément est notamment une caractéristique de l'approche système qui se veut en effet fondamentalement une façon de penser les systèmes en se positionnant toujours à haut niveau et en privilégiant donc les visions synthétiques globales aux visions analytiques locales (voir le paragraphe 2.2 ou l'excellent ouvrage [22] pour plus de détails sur ce sujet).

Au cœur du paradigme systémique se trouve bien sûr le concept de *système* que l'on définit classiquement comme un ensemble de composants (qui sont eux mêmes récursivement des systèmes), organisés et interagissant ensemble pour participer en permanence à une même mission (cf. chapitre 3 pour plus de détails), ce qui implique mécaniquement la capacité (plus ou moins grande) de tout système à pouvoir évoluer dans le temps pour réaliser sa finalité. On notera tout particulièrement que de nombreux systèmes n'ont pas de caractère industriel [4] car ils sont « simplement » le fruit de la nature – écosystèmes naturels, systèmes biologiques (espèce vivante, organe, cellule, etc.), systèmes physiques (univers, planète, atmosphère, volcan, matière, atome, etc.) – ou la résultante de l'activité humaine – écosystèmes artificiels (pays, ville, transport aérien, champ de bataille, etc.), organisations (entreprise, aéroport, système de distribution du courrier, système législatif ou administratif , etc.).

Comme nous ne pouvons pas encore définir vraiment précisément un système à ce stade de notre exposé, nous espérons cependant que cette notion pourra commencer à être mieux cernée par le biais de l'approche classificatoire que nous venons de prendre ci-dessus. On pourra cependant noter que tous les systèmes mentionnés plus haut sont toujours caractérisés :

- d'une part, par le fait qu'ils sont *formés de nombreux composants en interaction permanente* (ce qui fait qu'un système n'est en général jamais immobile, mais bien en mouvement et/ou en évolution permanente dès qu'on l'analyse de manière fine [5]),
- d'autre part, par leur *dimension intégrative*, i.e. par le fait que la résultante de toutes les dynamiques individuelles des éléments formant ces systèmes est une dynamique globale homogène et cohérente par rapport à une finalité d'ensemble,

ce qui illustre bien la définition très rapide d'un système donnée plus haut.

2. Le paradigme architectural

2.1 Le concept d'architecture

L'approche système est elle même sous tendue par un paradigme *architectural* que nous allons maintenant essayer d'expliciter plus en détails tant il est central dans cette démarche ! Pour bien comprendre cette nouvelle notion, rappelons d'abord que l'une des principales difficultés en matière de conception de systèmes industriels est liée au fait que les besoins à satisfaire et les propriétés attendues d'un système s'affinent progressivement au fur et à mesure de l'avancement de son ingénierie et ne se stabilisent le plus souvent que très tardivement dans le cycle de développement système. Le concepteur d'un système est donc obligé de travailler dans un environnement extrêmement mouvant où tout évolue et peut être

[4] Le terme « système industriel » renvoie ici aux nombreux systèmes fabriqués par l'industrie : systèmes de transport (automobile, avion, train, etc.), systèmes énergétiques (centrales nucléaires, centrales électriques, etc.) systèmes électroniques, logiciels, systèmes de télécommunications, systèmes d'information, etc.

[5] Tout dépend bien entendu du périmètre d'analyse que l'on prend : vue du système organisationnel « gratte ciel », une vitre est par exemple un composant terminal de bas niveau qui n'a simplement aucune dynamique propre (mais qui remplit une fonction importante pour les occupants du gratte-ciel) ; vue du physicien des matériaux, la même vitre sera au contraire un système plutôt complexe reposant une architecture cristalline et sur des échanges énergétiques permanents au niveau moléculaire et atomique.

remis en cause en permanence. Qui plus est, à ces difficultés classiques de conception, peuvent s'ajouter des difficultés d'analyse induites typiquement tant par le caractère souvent intrinsèquement distribué des systèmes considérés que par leur complexité intégrative. Ces différentes problématiques – souvent fortement liées à la maîtrise de la *dimension temporelle* de l'ingénierie et du fonctionnement des systèmes industriels – rendent de fait leur compréhension, leur conception et leur supervision extrêmement difficile [6] .

L'analyse architecturale d'un système est une réponse – la réponse par excellence, pourrions nous même dire – à ces problèmes structurels de maîtrise des systèmes : *l'architecture d'un système* peut en effet se définir comme la partie invariante de ce système, c'est-à-dire celle que l'on peut raisonnablement considérer comme fixe au cours du temps. C'est donc aussi celle qui va servir naturellement de point d'appui à la conception d'un système et celle sur laquelle on peut agir pour gérer l'évolution d'un système donné. Cela explique pourquoi, le terme « architecture » possède ici en pratique deux sens à la fois distincts, mais totalement corrélés : à savoir, d'une part, une dimension statique – les invariants d'un système – que nous avons pris ici comme définition et, d'autre part, une capacité dynamique, i.e. celle de concevoir et de faire évoluer un système. L'architecture d'un système – au sens premier donné plus haut – est donc un concept fondamentalement abstrait. Dans un environnement systémique où tout bouge au cours du temps, les seuls invariants naturels d'un système sont en effet les positionnements relatifs des différents systèmes (tant internes qu'externes) qui interagissent avec le système principal que l'on considère : trouver les invariants d'un système donné consiste de fait typiquement d'une part à comprendre la structure, la dynamique et les états possibles des composants de ce système et d'autre part à identifier l'ensemble des interfaces et des frontières – tant internes qu'externes – que possède ce système. On notera que cette vision statique de l'architecture est bien celle qui sous-tend sa dimension dynamique portée par l'architecte système : ce dernier est en effet avant tout celui qui définit l'organisation d'un système sous tous ses aspects, en jouant et en faisant évoluer notamment ses frontières internes et externes pendant sa phase de conception.

On remarquera tout particulièrement que cette manière d'étendre aux systèmes le concept d'architecture tel qu'il existe traditionnellement en urbanisme est parfaitement cohérente avec son acception usuelle. L'architecture d'un bâtiment n'est en effet autre que sa partie invariante, c'est-à-dire celle qui ne va pas changer quelle que soit la façon dont le bâtiment est occupé et évolue au cours du temps [7] . Il est clair que concevoir cette architecture, autrement dit faire les plans du bâtiment qu'on veut construire, revient exactement à définir l'organisation interne des éléments qui vont constituer ce bâtiment, cette organisation ne prenant par ailleurs son sens que par rapport à l'environnement où la nouvelle construction s'intègre : on retrouve ainsi également ici le fait que le cœur d'une démarche architecturale en urbanisme consiste à faire le choix des positionnements relatifs des constituants internes et externes de l'architecture d'un bâtiment, c'est-à-dire à définir leurs frontières. Pour se convaincre définitivement – s'il le fallait encore – que l'architecture au sens usuel de ce terme en urbanisme a les mêmes bases que celles de l'architecture système telles que nous avons essayé de les définir plus haut, nous invitons enfin le lecteur à méditer simplement les deux citations suivantes de Le Corbusier (1923 ; [14]), qui décrivent ce qu'est l'architecture d'une construction par opposition à son ingénierie et qui s'appliquent quasiment sans modification à l'architecture et à l'ingénierie des systèmes :

---

[6] Et parfois même impossible : il suffit de penser au *système financier mondial* pour voir à quel point un système artificiel peut ne pas être contrôlable quand il est complexe, alors même que sa dynamique est produite par des opérations élémentaires très simples (ici des ordres d'achat ou de vente, i.e. des additions ou des soustractions).

[7] Dans son célèbre traité « Vers une architecture » [14], Le Corbusier parle d'ailleurs de l'architecture comme de « cette chose qui dure à travers le temps », i.e. la partie d'un bâtiment qui est invariante au cours du temps !

- « l'ingénieur, conduit par *le calcul*, nous met en accord avec les lois de l'Univers ; il atteint *l'harmonie* » ;
- « l'architecte, par *l'ordonnance des formes*, […], par *les rapports* qu'il crée, […], nous donne la mesure d'un ordre qu'on sent en accord avec celui du monde […] ; c'est alors que nous ressentons *la beauté*. »

2.2 Paradigme architectural versus paradigme analytique

Il est aussi important de comprendre que la pensée architecturale s'oppose d'une certaine manière à la pensée analytique classique. Raisonner en architecte, ce n'est en effet absolument pas chercher à comprendre dans le moindre détail comment va fonctionner le système dont on s'occupe (ce qui serait ce à quoi s'attacherait une approche d'ingénierie analytique), mais bien plus à identifier quels vont être les grands invariants structuraux du système pour laisser ensuite le système trouver son propre équilibre dans le cadre architectural qu'on lui aura donné (et qu'il devra structurellement préserver et respecter).

	Paradigme analytique	**Paradigme architectural**
Principe fondateur	Compréhension complète	Compréhension globale
Périmètre d'action	Systèmes homogènes	Systèmes hétérogènes
Substrat théorique	Discipline scientifique	Logique & sémantique
Mode de pensée	Certitude	Relativité
Mode de représentation	Représentation détaillée	Points de vue
Interactions projets [8]	Locales (expertise)	Globales (collaboratif)
Spécialiste industriel	Ingénieur	Architecte

Tableau 1 – Comparaison raisonnée des paradigmes analytique et architectural

Cette opposition entre pensée analytique et démarche architecturale est en fait exactement de la même nature que celle qui structure les mathématiques avec le distinguo fondamental entre grosso modo, analyse, probabilités & mécanique d'une part, et algèbre, logique & géométrie d'autre part : de même, l'ingénieur analyste est celui qui manipule les équations et les calculs tandis que l'architecte géomètre est celui qui joue avec les relations et les formes. A partir de là, une deuxième différence assez fondamentale entre ces deux modes de pensée va apparaître tout à fait naturellement : d'un côté, la recherche de la compréhension complète d'un système portée par la pensée analytique fait qu'elle ne s'applique en fait efficacement que dans des

[8] Au niveau industriel, l'architecture a fondamentalement pour objectif la maîtrise des processus d'*intégration*, c'est-à-dire des processus de conception et de fabrication de systèmes (globaux) par mise en relation d'un ensemble hétérogène de sous-systèmes (en général homogènes quand on les considère individuellement). La conception de l'architecture de tels systèmes s'appuie donc sur une nécessaire collaboration entre les différents experts qui portent la conception de chacun des sous-systèmes qui forment le système global qu'on cherche à intégrer : l'architecte système doit notamment coordonner dans une logique d'ensemble toutes les parties prenantes du système global pour que les choix locaux d'architecture, i.e. ceux qui vont être fait localement au niveau de chacun des sous-systèmes du système considéré, soient toujours cohérents avec les choix globaux d'architecture, i.e. ceux qui se font au niveau le plus haut du système global pris dans son ensemble.

contextes très homogènes en mettant en œuvre un corpus disciplinaire bien défini, ce qui conduit à un mode de pensée d'expert dominée par la *certitude* ; de l'autre côté, la démarche architecturale cherche à obtenir une vision globale d'ensemble d'un système, ce qui se fait nécessairement au détriment d'une vision plus fine et ne s'applique avec pertinence que dans des contextes hétérogènes où il va falloir procéder par consolidation de points de vue très multiples, ce qui in fine débouche sur un mode de pensée « logique » marqué par une très profonde *relativité* [9] : toute frontière – l'objet clef avec lequel l'architecture joue – n'a par exemple typiquement pas d'existence intrinsèque dans la mesure où elle résulte toujours d'une décision – i.e. un choix d'architecture – ayant un certain degré d'arbitraire et des contraintes croisées existant entre les deux composants architecturaux qu'elle sépare.

Bien que nous venons d'opposer les modes de pensée analytique et architecturale pour mieux mettre en évidence la spécificité du paradigme architectural, il faut cependant se rendre compte que ces démarches sont en fait totalement complémentaires en pratique. Un architecte doit notamment typiquement être capable de faire appel en permanence au meilleur de ces deux paradigmes en fonction de ses besoins [10] : lorsque ce dernier doit approfondir la pertinence d'une solution architecturale et qu'il est confronté à des problématiques de dimensionnement, c'est bien par exemple une démarche de nature analytique qui typiquement lui permettre de qualifier sa solution.

2.3 Les enjeux de l'architecture système

Pour conclure ce chapitre, il nous paraît extrêmement important de re-souligner le pourquoi de l'architecture système. Cette discipline a de fait pour objectif fondamental de fournir des leviers structurants pour maîtriser les systèmes industriels – c'est-à-dire pour être capable de les concevoir, de les fabriquer et de les mettre en œuvre sous *contraintes de qualité, de coût, de temps et de performance* (QCDP) – lorsqu'on se trouve des environnements complexes [11] .

L'architecture système apporte donc de ce fait logiquement des outils et des techniques pour attaquer de front les trois grandes catégories de problèmes suivants qui minent classiquement la qualité, le coût, les délais de fabrication et la performance des systèmes industriels :

- *le manque de recul systémique :* beaucoup de projets systèmes voient leurs métriques QCDP exploser en raison de problèmes de fond qui sont détectés bien trop tard dans le cycle de conception et qui conduisent typiquement à repenser l'architecture globale des systèmes concernés, engendrant alors classiquement des augmentations de coût et des délais supplémentaires ainsi que des problèmes de qualité voire de performance ; l'approche architecturale apporte ici une réponse de bon sens – malheureusement trop peu souvent mise en oeuvre – à cette problématique, qui consiste « simplement » à essayer d'avoir le plus en amont possible du cycle de développement la vue la plus

[9] L'architecture d'un système n'est notamment le plus souvent pas « bonne » de manière intrinsèque, dans la mesure où elle dépend de nombreux critères systémiques. Une architecture n'est en effet qu'une solution – en général parmi beaucoup d'autres possibles – qu'on va choisir en fonction de critères extrinsèques à l'architecture à proprement parler. Tout l'art de l'architecte consiste justement à expliciter ces critères pour que les commanditaires du système puissent faire – en toute connaissance de cause – les choix stratégiques qui sont de leur ressort (e.g. les choix de budget, de niveau de qualité, de niveau de sécurité, etc.). Selon la nature de ces choix, on peut bien sûr obtenir des architectures radicalement différentes pour le système considéré.

[10] Savoir jongler entre les deux paradigmes n'est néanmoins pas chose facile et devrait normalement s'apprendre dès le plus jeune âge. Cela étant dit, il reste encore du chemin à parcourir pour imposer l'architecture comme un savoir-faire fondamental, qui devrait être acquis par tout ingénieur dans sa formation initiale (surtout en France où la culture Cartésienne traditionnelle a beaucoup de mal à penser l'évolutivité extrinsèque des systèmes !).

[11] L'architecture système est donc fondamentalement au service de la maîtrise « business » des grands systèmes.

claire possible du système à réaliser *dans sa globalité* ; on s'appuie notamment à cet effet sur une modélisation à « haut niveau » de l'ensemble du système à réaliser, pour se donner de la hauteur de vue et résoudre tous les problèmes systèmes avant de se focaliser sur les détails (qui absorbent très souvent les ingénieurs en pratique, sans que pour autant les grands problèmes de fond soient réglés).

- *la barrière de la complexité :* quand un système industriel est obtenu par intégration d'autres systèmes, ces derniers résultant récursivement eux-mêmes du même type de mécanisme et que la hiérarchie systémique correspondante est de taille importante, *sa complexité* – que ce soit en terme de variables systèmes, de nombre d'interfaces, de technologies utilisées, d'effets émergents, de gestion de projet, etc. – est extrêmement difficile à maîtriser en pratique comme nous l'avions déjà souligné au paragraphe 1.1 de cet article ; l'architecture système apporte à nouveau quelques outils pour essayer de faire face à cette problématique dont le plus important est sans doute le processus d'abstraction (cf. [3]) : ce dernier consiste typiquement à dégager systématiquement le « bon » grain d'analyse de haut niveau d'un système en évitant notamment de se perdre dans les détails techniques qu'il convient toujours d'abstraire dans un premier temps ; cela revient donc exactement à travailler avec la représentation la plus simple possible du système à haut niveau, sans pour autant être simpliste, ce qui n'est jamais chose facile ; la logique architecturale sous-jacente est par ailleurs assez simple à comprendre : il est de fait difficile d'imaginer un système qui soit convenablement architecturé à un niveau fin sans qu'il le soit d'abord à haut niveau !

- *la mauvaise maîtrise des interfaces :* l'essentiel des problèmes d'intégration est lié à la gestion des interfaces [12] tant internes qu'externes ; il est en effet fondamental de pouvoir *découpler* au maximum les sous-systèmes qui forment un système pour éviter que l'évolution d'un sous-système donné ne « pollue » les autres sous-systèmes avec lesquels il est interfacé, ce qui est un facteur classique de difficulté à la fois en terme de conception et de maintenance ; ces problèmes d'interfaces sont bien sûr au cœur de la démarche architecturale qui va notamment s'attacher à commencer d'abord par « bien » comprendre un système dans son environnement pour avoir la vision la plus claire possible de ses interfaces externes avant de se plonger dans l'optimisation de ses interfaces internes ; on notera aussi que la méthodologie architecturale consiste typiquement *à identifier, à partager et à négocier* chaque interface car une interface technique cache bien entendu (presque) toujours des interfaces « humaines » dans le cas des systèmes industriels (voir paragraphe 4.1 pour plus de détails).

Comme on le voit, bien qu'apparemment très abstraite, la démarche d'architecture système a donc un objectif extrêmement concret qui la structure en profondeur, à savoir l'élimination de tous les facteurs de dégradation de l'efficacité (au sens QCDP) d'un système.

3. Les outils de l'architecture système

3.1 La description d'un système

Pour mieux comprendre ce qu'est l'architecture système, nous allons d'abord présenter plus en détails l'objet principal qu'elle manipule, à savoir le concept central de *système*. Notons cependant que l'acception que nous allons donner à ce terme est relativement personnelle (par

[12] Une interface entre deux systèmes A et B est la donnée des contraintes logiques croisées C(A,B) et C(B,A) que les systèmes A et B font respectivement peser l'un sur l'autre. Il est donc important de comprendre qu'une interface en ce sens n'a aucun réalité matérielle : gérer une interface consiste notamment « juste » à garantir que la conjonction logique de C(A,B) et de C(B,A) est toujours vraie au cours du temps !

rapport au contexte de l'ingénierie système) dans la mesure où elle se veut mathématiser la définition totalement informelle de l'INCOSE. Rappelons ici au passage que cette dernière définit récursivement un système comme un ensemble organisé d'autres systèmes (matériels, logiciels et humains) de manière à ce que leur intégration permette au système d'accomplir – dans un environnement donné – la mission pour laquelle il a été conçu.

Définition 3.1 – Système formel – *Un* système formel *est la donnée d'une fonction partielle F de transformation de flots* [13] *de la forme :*

$$y(t) = F(x(t) ; q(t) ; t)$$ [14]

où t varie dans un ensemble (discret ou continu) $\mathbb{T}$ qui modélise le temps.

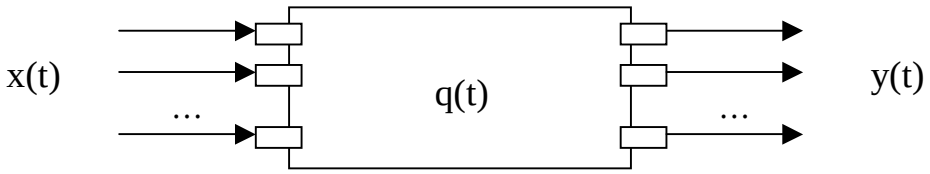

Figure 2 – Représentation symbolique d'un système formel

Pour construire de nouveaux systèmes à partir d'autres systèmes déjà connus, on peut faire appel à deux types d'opérateurs qui jouent un rôle fondamental en architecture système :

- l'opérateur d'*intégration* : cet opérateur *I* consiste à prendre un ensemble de systèmes formels interfacés de façon cohérente les uns avec les autres F_1, …, F_n (voir Figure 3) et à fabriquer un nouveau système formel $I(F_1, …, F_n)$ grâce à un mécanisme de « boite noire » consistant à ne voir que le comportement entrée / sortie global de l'ensemble des systèmes que l'on considère et qui est induit par leurs entrées libres (la sémantique de cet opérateur est une sémantique de point fixe ; cf. [10] ou [11]) ;

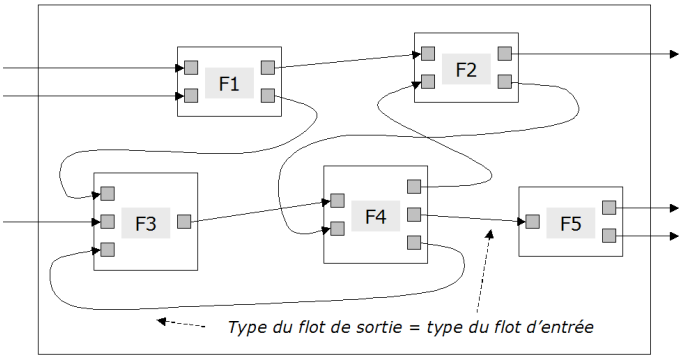

Figure 3 – Exemple d'un opérateur d'intégration

- les opérateurs d'*abstraction/concrétisation* : ces opérateurs *A* et *C* sont construits avec des opérateurs classiques d'abstraction/concrétisation α et χ au sens de l'interprétation abstraite (cf. [3] par exemple) portant sur les « grandeurs » transportées par les flots

[13] C'est-à-dire de suites indexées par $\mathbb{T}$ d'éléments d'un ensemble partiellement ordonné $\mathbb{P}$ donné (i.e. de $\mathbb{P}^{\mathbb{T}}$).

[14] x(t), y(t) et q(t) sont respectivement appelés les flots d'entrée, de sortie et des états du système considéré.

d'un système formel F [15] – qu'on étend naturellement entrée par entrée aux flots – en définissant alors respectivement l'abstraction et la concrétisation d'un système formel F comme $A(F) = \alpha \circ F \circ \chi$ et $C(F) = \chi \circ F \circ \alpha$ (voir Figure 4).

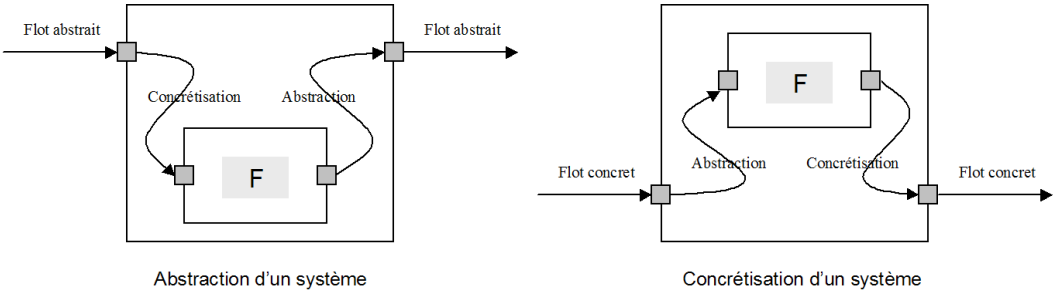

Figure 4 – Abstraction et concrétisation d'un système formel

Le processus d'analyse d'un système industriel consiste alors typiquement à pouvoir identifier récursivement (jusqu'à arriver à des systèmes que l'on considère comme totalement connus et maîtrisés) un tel système réel à un système formel du type :

$$F = I(A(F_1),\dots,A(F_n)) \quad (1),$$

que l'on voit donc comme une intégration d'abstractions de systèmes formels, correspondant aux « sous-systèmes » du système étudié. Le niveau « système » est alors naturellement celui des flots abstraits d'un tel système formel alors que le niveau « sous-système » s'identifie à celui des flots concrets manipulés par les sous-systèmes qui le compose.

On notera que notre approche se veut suffisamment générique pour rendre compte de tous les systèmes industriels auxquels on peut être confronté en pratique. Le travail d'un architecte système peut donc s'analyser en toute rigueur comme un travail de modélisation d'un système réel dans le cadre formel que nous venons de définir. En pratique, on n'utilise bien entendu que rarement des formalismes aussi abstraits que celui que nous venons de définir. Cela étant dit, ce cadre théorique permet maintenant facilement d'identifier les éléments « minimaux » qui permettent de représenter l'architecture d'un système industriel d'une façon équivalente au formalisme précédent. Ce sont ces derniers modèles de représentation – que nous allons maintenant introduire – qu'on retrouve notamment à la base de la quasi totalité des langages de description architecturale (dont SysML est un bon exemple représentatif ; cf. [9] ou [27]).

Si on veut disposer d'une représentation (récursive) complète d'un système modélisé dans le cadre formel qui vient d'être introduit, en supposant que l'on dispose récursivement d'une description de chacun des « sous-systèmes » qui le constitue, on voit en effet immédiatement (en mettant bout à bout la Définition 3.1 et l'équation (1)) qu'il faut pouvoir décrire à la fois :

- les *éléments constitutifs de la fonction de transfert* du système étudié, ce qui nécessite donc de pouvoir décrire :

  → le type des flots abstraits manipulés par le système, i.e. le *modèle temporel* (discret ou continu) associé à chaque flot et la *nature des « grandeurs »* transportées par les flots (que l'on appelle également usuellement « objets métiers »), ce qui se fait typiquement par le biais d'un *modèle de données* (cf. Figure 5),

[15] I.e. l'ensemble partiellement ordonné sous-jacent au flot (voir la note de bas de page 13).

→ les *variables d'états* du système, ce qui peut naturellement se faire en utilisant une *machine à états finis* (automate fini) qui permet aussi de préciser la dynamique des changements d'états (i.e. les événements qui conduisent à changer d'état) en même temps que la nature des états du système (cf. Figure 5),

→ les *conditions* sur les variables d'entrée et de sortie (et éventuellement les états) du système qui restreignent son champ de validité (et donne un caractère partiel à sa fonction de transfert), qui forment donc autant d'*exigences sur le comportement observable du système* qui doivent être vérifiées par ce dernier (cf. Figure 5),

- la *relation d'intégration* donnée par l'équation (1), ce qui nous amène donc à savoir décrire les éléments suivants :

→ les *paramètres syntaxiques* de l'opérateur d'intégration, i.e. la liste de l'ensemble des systèmes en jeu ainsi que la relation système / « sous-système » induite, qui se décrit typiquement dans *un diagramme de bloc* en SysML (cf. Figure 5),

→ la *fonction définie par l'opérateur d'intégration*, qui nécessite de décrire d'une part la *structure syntaxique* de cet opérateur, i.e. les relations fonctionnelles liant les différents « sous-systèmes » en jeu et d'autre part le *comportement temporel dynamique* induit par cet opérateur, autrement dit l'algorithmique inter-« sous-systèmes », ce que l'on doit d'ailleurs souvent faire à état fixé [16] (cf. Figure 5),

→ les *relations d'abstraction* intervenant dans (1), dont on peut rendre compte dans le *modèle de données* déjà utilisé pour décrire les flots du système considéré, en y reportant tant les grandeurs concrètes manipulées par ces relations d'abstraction que leurs relations avec les objets métiers déjà présents (cf. Figure 5).

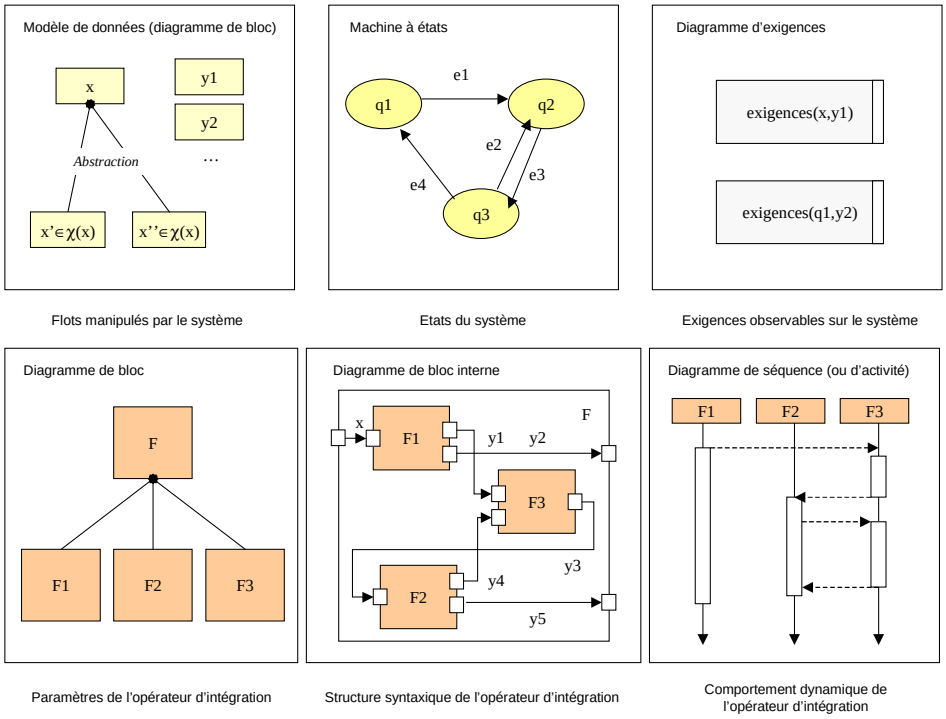

Figure 5 – Les types de diagrammes nécessaires et suffisants pour représenter un système

[16] La structure syntaxique (resp. le comportement dynamique) d'un opérateur d'intégration se décrit en SysML à l'aide typiquement d'un diagramme de bloc interne (resp. de séquence ou d'activité).

On voit ainsi que tout système peut récursivement se décrire avec un nombre limité de types de diagrammes qui existent notamment tous en SysML – qui a donc une richesse d'expression suffisante pour représenter tout système – dont nous reprenons ici la terminologie ([9], [27]) : le diagramme de bloc (systèmes et objets métiers), la machine à états (états), le diagramme de bloc interne (structure syntaxique d'un opérateur d'intégration), le diagramme d'activité ou de séquence (comportement dynamique induit par un opérateur d'intégration) et le diagramme d'exigences (conditions sur le comportement observable d'un système).

3.2 Les référentiels d'architecture

Comme nous venons de le voir au paragraphe précédent, l'approche architecturale manipule de nombreux types de représentations d'un système car l'on ne saurait raisonner – et plus généralement agir – sur un système industriel en cours de conception sans description de ce système. Pour des raisons liées fondamentalement aux processus industriels de conception, on sépare classiquement les types de représentation qui ont été introduits à la fin du paragraphe précédent (cf. Figure 5) en deux grandes catégories qui structurent le référentiel de l'ingénieur dans la démarche système (cf. Figure 6), à savoir :

- les *exigences* qui ne sont autres que des conditions « logiques » qui doivent toujours être satisfaites par le système que l'on veut construire,
- les *spécifications* qu'on peut définir comme un ensemble non ambigu de descriptions du système que l'on cherche à réaliser (i.e. ses « plans » en quelque sorte).

Le lecteur averti se rendra cependant compte que cette subdivision n'est pas complètement opérante dans le formalisme de description de systèmes qui a commencé être à introduit : il n'est en effet pas possible de définir de manière non ambiguë un système sans être capable d'exprimer aussi certaines conditions logiques portant sur les paramètres structurants de ce système, comme cela résulte immédiatement du travail d'analyse du paragraphe précédent [17] .

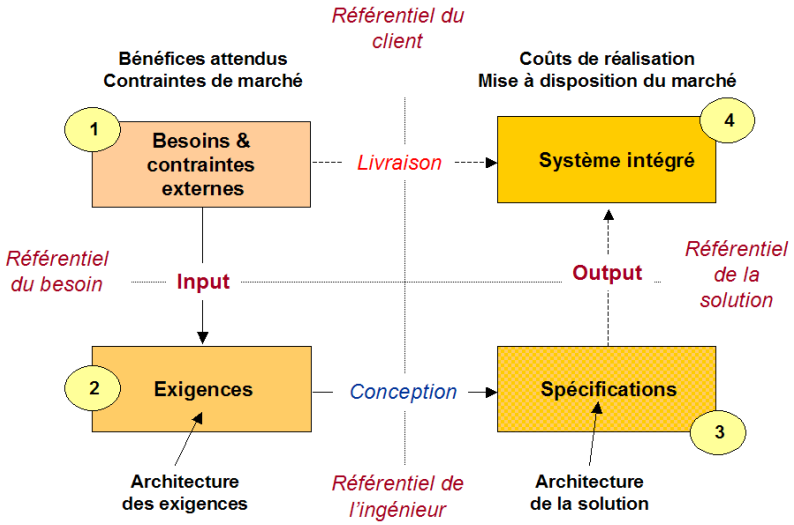

Figure 6 – Les référentiels classiques de l'architecture système

[17] Il est notamment impossible d'exprimer des propriétés systémiques du style dans des formalismes autres que logiques : « la relation P(x, …, y) entre les variables x, …, y doit toujours être vérifiée » (invariance) ou « quand le système est dans l'état q, il doit toujours repasser au bout d'un certain temps par l'état q' » (vivacité).

La frontière entre exigences et spécifications n'est donc pas parfaitement claire ! Pour bien la percevoir, il faut l'analyser dans le cadre du processus de réalisation d'un système industriel (cf. Figure 6). Celui-ci débute en général par une expression de besoins, souvent pas très bien formalisée, provenant d'un client [18] . Le premier travail de l'architecte système se situe donc logiquement dans le même référentiel que celui de son client (*référentiel du besoin*) et va consister à structurer et à clarifier ces besoins clients en les transformant en exigences, i.e. en propriétés non ambiguës qui doivent être vérifiées par le système cible. Une fois que le besoin est bien compris, on peut alors passer dans le *référentiel de la solution* en attaquant la phase de conception / spécification qui va mécaniquement amener à remplacer certaines exigences issues de l'explicitation du besoin client par des représentations symboliques (similaires à celles introduites au paragraphe précédent) [19] pour aboutir à des spécifications formées à la fois de diagrammes et d'exigences « irréductibles » qu'on ne peut pas représenter à l'aide de formalismes graphiques. On voit donc que la séparation entre les concepts d'exigences et de spécifications se fait principalement au niveau des processus d'ingénierie sous-jacents : la phase d'ingénierie des exigences est en effet (relativement) séparée de la phase de conception / spécification à laquelle elle fournit un livrable d'entrée qui est une architecture d'exigences que l'on retrouvera donc partiellement dans l'architecture de la solution [20] .

3.3 Les points de vue architecturaux

Pour analyser un système réel donné, l'architecte système utilise classiquement trois angles d'analyse – ou visions (cf. Tableau 2) – qui structurent notamment le référentiel de l'ingénieur (au sens du paragraphe précédent), i.e. les exigences et les spécifications, à savoir :

- la vision *opérationnelle* qui a pour but de définir le *pourquoi* du système, autrement dit de préciser la mission du système et ce à quoi sert le système,
- la vision *fonctionnelle* qui a pour objectif d'expliciter *le fonctionnement l*ogique du système, i.e. ce qu'il fait indépendamment de la façon dont on le réalisera,
- la vision *organique* qui définit la façon dont le système est *concrètement réalisé*, i.e. l'organisation et la dynamique de ses composants matériels, logiciels et humains.

Ces trois visions architecturales sont bien entendu à replacer dans le contexte du processus de modélisation systémique standard (voir paragraphe 4.2) où l'architecte système modélise son système cible à l'aide d'un mécanisme de concrétisation progressive [21] consistant à modéliser

[18] On notera que le terme « client » ne fait nécessairement référence ici à l'utilisateur ou au client final. Il peut en effet s'agir également d'un client interne (typiquement le département marketing dans le cas des entreprises – comme l'automobile ou les télécoms – qui mettent des systèmes sur le marché en mode « push »).

[19] Typiquement un exigence du type « le système doit être capable de faire la fonction F » va se représenter en mettant simplement un bloc modélisant la fonction « F » dans un diagramme ad hoc.

[20] Cette séparation classique a cependant sans doute vocation à s'affaiblir dans le futur pour converger à terme sur une seule phase de conception systémique où toutes les représentations du système en cours de réalisation (exigences et représentations symboliques) sont construites en même temps dans la droite ligne de l'analyse faite au paragraphe précédent : c'est typiquement la logique sous-jacente à des formalismes comme SysML où l'on manipule dans un même environnement à la fois des diagrammes d'exigences et de structures.

[21] On notera que le cœur du concept de vision architecturale est lié à ce mécanisme de concrétisation progressive. Il est en effet parfois nécessaire d'introduire d'autres visions que celles que nous avons introduites pour « bien » rendre compte de tel ou tel type de système (typiquement les systèmes d'information ou les systèmes à logiciel dominant où il peut être respectivement utile de disposer de visions « services » et/ou « données »). Si l'on veut être suffisant large pour capturer tout type de situation, on pourrait donc typiquement définir un ensemble de visions architecturales d'un système comme la donnée d'une suite de systèmes formels (F_i) qui sont tous des

successivement la dimension opérationnelle, fonctionnelle et enfin organique de son système à l'aide de systèmes formels OP, FO, OR tels que OP = A(FO) et FO = A(OR) [22] .

On remarquera qu'on peut donner un sens rigoureux aux relations existant entre les visions architecturales d'un système, mais pas à chaque vision donnée. Le caractère propre de chaque vision reste ainsi une notion « non mathématisable » car on peut pas vraiment énoncer des critères extrêmement précis pour savoir si tel élément intervenant dans un modèle systémique est de nature opérationnelle, fonctionnelle ou organique. Nous sommes donc là en plein dans l'art de l'architecte et de la modélisation qui consiste à faire un pont entre le réel et le formel : classer un élément d'architecture dans l'une des trois visions est donc fondamentalement un acte de modélisation en ce sens là (et donc de caractère non formel).

On notera aussi que l'introduction de la notion de vision architecturale conduit naturellement à organiser un modèle systémique selon ces visions et donc à typer les éléments de description d'un système (cf. paragraphe 3.1) en fonction de la vision à laquelle ils se rattachent.

Visions	**Répond à la question**	**Exemples de points de vue**	**Instances de points de vue (brosse à dents électronique)**
Opérationnel	Pourquoi ?	Mission, contexte opérationnel, contexte stratégique	Dents propres et saines, gain de temps, salle de bain « tendance »
Fonctionnel	Quoi ?	Fonction, fonctionnement & mode de fonctionnement	Brossage, régulation de vitesse, programmation de la force de brossage
Organique	Comment ?	Composant, organe & configuration technique	Tête, base, corps, régulateur de vitesse

Tableau 2 – Les visions et les points de vue en architecture système

Signalons enfin que ces visions architecturales peuvent encore se raffiner selon d'autres axes d'analyse qui donnent naissance à des points de vue plus fins. La méthode SAGACE (cf. [20], [18] et/ou [19]) propose par exemple une grille où le second axe de lecture est la nature de la dynamique temporelle de la relation entre un système industriel et son système de supervision (quand il existe) ce qui conduit à segmenter les trois visions architecturales par rapport aux actions (comportements instantanés du système), aux réactions (comportements du système résultant d'une action de contrôle de premier niveau) et aux pro-actions (comportements du système résultant d'une action de contrôle de second niveau). Les points de vue détaillés qui résultent de cette grille d'analyse sont explicités dans le Tableau 2. Il existe cependant aussi

abstractions d'un même système cible F (i.e. tels que F_i = A*(F)* pour tout *i*), puis instancier le type de systèmes intermédiaires dont on a besoin en fonction du contexte de modélisation concerné.

[22] Signalons au passage qu'apparaît maintenant clairement une difficulté intrinsèque au processus d'architecture système car on voit bien que l'architecte système doit manipuler en pratique deux types d'abstraction, à savoir celles qui définissent la hiérarchie systémique (la relation récursive système – sous-systèmes) et celles qui sont liées aux visions architecturales. Le « modèle » systémique qui en résulte est donc un hyper-cube à plusieurs axes dans lequel l'architecte doit se déplacer en permanence lorsqu'il modélise son système …

d'autres axes d'analyse qui conduisent à des points de vue différents. On peut par exemple analyser un système par rapport au modèle de système formel de la Définition 3.1, ce qui conduit à travailler avec les axes d'analyse donnés respectivement par sa structure statique, son comportement dynamique, la loi d'évolution de ses états et la nature des données qu'il manipule. On aboutit donc alors aux points de vue déjà décrits dans la Figure 5 qui sont ceux que l'on peut typiquement facilement manipuler dans un langage de modélisation systémique comme SysML (cf. à nouveau [9] ou [27]).

4. La mise en œuvre d'un processus d'architecture système

4.1 Système technique et système organisationnel

Comme nous l'avons déjà souligné au paragraphe 1.2 (cf. Figure 1), les systèmes industriels sont caractérisés par le fait qu'ils résultent d'un mécanisme de conception et de fabrication géré par une organisation industrielle (d'où leur dénomination). Il est donc important de bien comprendre qu'un système technique n'existe jamais seul et que le premier travail de tout architecte système (en appliquant les principes mêmes de l'architecture système au système « système technique – organisation industrielle » dont il est partie prenante) est d'identifier l'architecture organisationnelle où il évolue et la manière dont elle s'accoste à l'architecture technique du système qu'il doit concevoir (cf. Figure 7). Le travail d'un architecte système est de ce fait logiquement de nature double et consiste à la fois en :

- un *travail technique* qui est centré fondamentalement sur la définition des interfaces techniques des sous-systèmes qui composent le système dont il a la responsabilité [23] ,
- un *travail de facilitation* qui est centré sur la convergence des parties prenantes de ces sous-systèmes techniques sur ces choix d'interfaces.

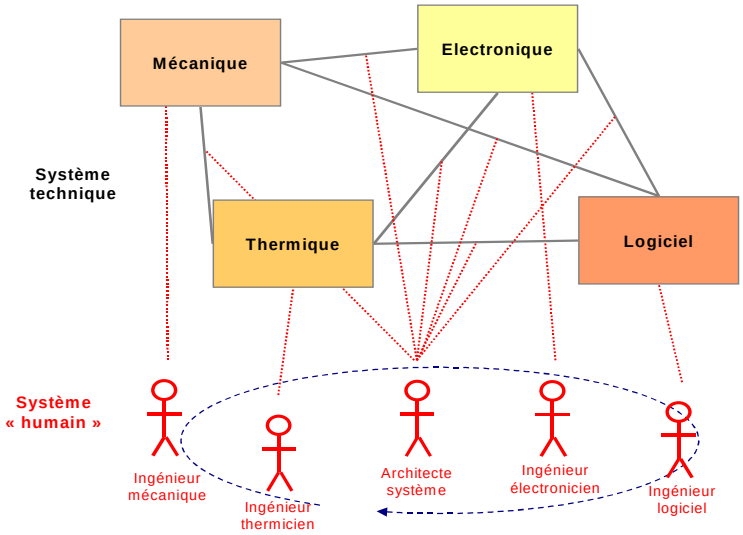

Figure 7 – Les deux systèmes technique et humain en jeu dans un projet système

[23] Toute la difficulté de la compréhension de l'importance de l'architecture système vient de cette situation. Les ingénieurs s'occupent des sous-systèmes d'un système (i.e. les « boites » dans la Figure 7) qui sont clairement visibles de tous alors que l'architecte s'occupe de ses interfaces (c'est-à-dire les « flèches » dans la Figure 7) que personne ne voit car elles ne sont pas matérielles stricto sensu. Le travail architectural se fait ainsi quelque part dans l'invisible et est donc difficile à percevoir pour le non initié alors qu'il fondamental puisqu'il fixe le cadre de travail de l'ingénierie (un mauvais cadre ne saurait typiquement que conduire à un « mauvais » système).

Ces deux aspects indissolubles et totalement complémentaires du métier d'architecte système sont décrits plus en détail dans les deux paragraphes qui suivent.

4.2 Processus d'analyse et de modélisation des systèmes complexes

Le travail d'analyse technique de l'architecte système a pour double objectif de bien structurer l'architecture d'ensemble du système et d'identifier les interfaces de niveau système (i.e. les interfaces externes et les interfaces inter-sous-systèmes). Les deux phases clefs de ce travail – qui n'est ici considéré qu'au seul niveau système – sont donc les suivantes (cf. Figure 8) :

- l'identification du *périmètre systémique* du système qui a pour but d'identifier tous les systèmes externes – et d'analyser au passage la nature fine des *interfaces externes* correspondantes – qui ont une influence suffisamment notable sur le système cible pour qu'on doive en tenir compte dans son processus de conception [24] ,
- l'obtention successive des architectures *opérationnelle, fonctionnelle et organique* du système [25] qui va donc déboucher sur la compréhension fine de la structure interne du système – et notamment de ses *interfaces internes* – sur les axes d'analyse donnés par les trois visions architecturales (cf. paragraphe 3.3).

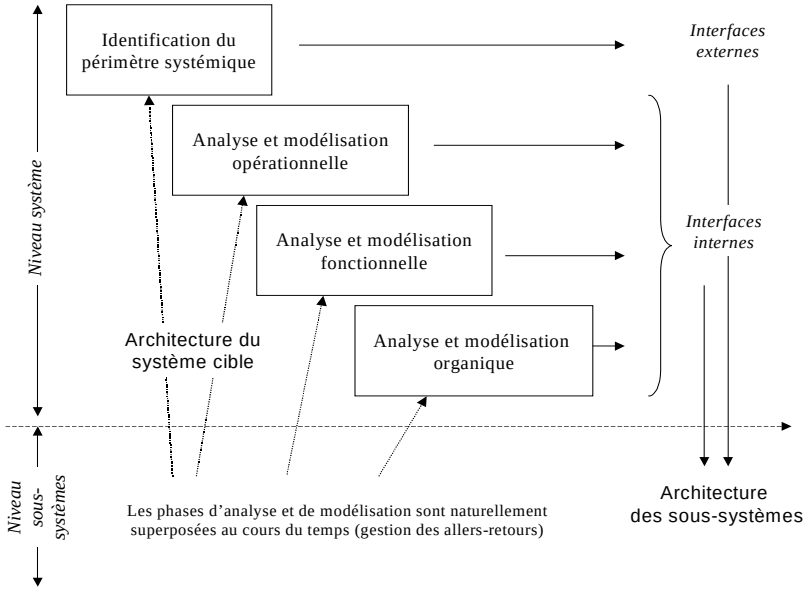

Figure 8 – Le processus (récursif) d'analyse et de modélisation systémique

Dès que ces éléments sont obtenus, on peut démarrer récursivement les phases de conception des sous-systèmes du système cible. Celles-ci commencent typiquement lorsque l'architecte

[24] Identifier le périmètre systémique d'un système donné revient à trouver un sur-système trivial T (c'est-à-dire formellement sans entrées / sorties) du système considéré et à décomposer ce sur-système sous la forme d'une intégration – au sens de l'équation (1) – du système étudié et d'autres systèmes (dits externes). Cela revient en effet à considérer que les autres systèmes (en dehors de T) n'ont pas d'influence sur le système cible, ce qui fait qu'on peut également négliger leur influence sur les sous-systèmes constitutifs de T (i.e. le système analysé et ses systèmes externes), et donc n'appartiennent pas au périmètre systémique du système que l'on analyse.

[25] Ce qui revient formellement à expliciter l'équation (1) introduite au paragraphe 3.1 par rapport aux 3 visions opérationnelle, fonctionnelle et organique.

système passe la main aux architectes ou aux ingénieurs sous-systèmes [26] en leur donnant les spécifications des interfaces externes de chacun des sous-systèmes telles qu'elles résultent de son travail d'analyse et de modélisation.

Il est cependant important de souligner que ce travail de conception architecturale n'a en fait *aucun caractère linéaire* même si nous utilisons le terme de « phase » pour parler des grandes étapes qui le structurent. Il faut en effet bien voir que les éléments constitutifs d'un modèle systémique sont tous inter-reliés, ce qui fait qu'il est fréquent – pour ne pas dire naturel et plus que normal – d'être amené à retravailler et/ou à redéfinir une partie antérieure d'un modèle (qu'on aura oubliée ou mal modélisée compte tenu d'éléments qui seront apparus plus tard dans le travail d'analyse) alors qu'on a abordé une phase plus amont de modélisation. Le processus de modélisation systémique au niveau système se fait donc en pratique selon un axe principal « modélisation du périmètre systémique, architecture opérationnelle, architecture fonctionnelle, architecture organique », mais tout en gérant également en permanence des allers-retours entre ces différentes dimensions. Il se fait de même aussi selon un autre axe « système – sous-systèmes », mais on notera que sur ce plan, l'enjeu fondamental de l'analyse au niveau système est bien de minimiser au maximum les éventuels allers-retours potentiels qui pourraient amener à remettre en cause la modélisation système lorsqu'on analyse plus en détails chacun des sous-systèmes du système cible !

4.3 Processus de convergence des parties prenantes des projets systèmes

Pour rebondir sur la fin du paragraphe précédent, on voit donc qu'un des éléments clefs du travail de l'architecte système est de garantir que les interfaces des sous-systèmes du système cible qu'il conçoit soient raisonnablement robustes pendant le projet système [27] , autrement dit qu'elles ne soient pas (ou en tout cas le moins possible) remises en cause par les responsables techniques des sous-systèmes au cours d'un projet système. Pour arriver à cette situation, il est nécessaire de jouer sur une dimension différente que la seule dimension technique : tout le problème de la conception technique est en effet qu'il ne suffit pas d'avoir la « meilleure » architecture système possible pour qu'elle soit automatiquement reprise et utilisée par chacun des responsables sous-systèmes. L'optimum global, typiquement par rapport aux critères de qualité, de coût, de délais et de performance, d'un système intégré n'est en effet pas l'union des optimaux locaux de chacun des sous-systèmes qui le composent : il en résulte que chaque sous-système pris individuellement doit en général être sous-optimal par rapport à ces mêmes critères systèmes, ce qui n'est pas nécessairement facile à admettre pour les responsables de ces sous-systèmes car ils n'aborderont pas la conception de leurs sous-systèmes en ayant une vision globale des problèmes, comme peut l'avoir l'architecte de l'ensemble du système. Pour résoudre cette difficulté « humaine » consubstantielle à la conception des systèmes intégrés, il est nécessaire d'intégrer au cœur de tout processus d'architecture système des mécanismes d'alignement des parties prenantes de ce processus sur les choix structurants d'architecture au niveau système, ce qui revient – systémiquement parlant – exactement à s'assurer que toutes les interfaces techniques d'un système soient toujours partagées et acceptées au niveau des interfaces « humaines » du système projet auxquelles elles s'accostent (cf. Figure 7)

Cette analyse rapide montre donc qu'un architecte système doit fondamentalement toujours être capable de faire converger des acteurs sur les choix d'architecture dont il est responsable. Ce travail de convergence – qui est une partie intégrante du métier d'architecte système [28] –

[26] Selon le niveau de complexité des sous-systèmes en jeu.

[27] Et si possible au delà, mais là, c'est un autre débat, à savoir celui de la réutilisabilité des architectures et de la conception des gammes de produits, que nous n'aborderons pas ici.

[28] Et qui fait d'un « bon » architecte système une sorte de mouton à 6 pattes …

n'est cependant pas simple car il nécessite de maîtriser, au dessus de compétences purement techniques, un ensemble de compétences d'une nature totalement orthogonale, qu'on pourrait typiquement qualifier d'architecture et d'ingénierie humaine car elles nécessitent de travailler sur le plan purement « humain ». On peut ainsi par exemple citer à ce niveau la capacité :

- *d'identifier toutes les parties prenantes* d'une architecture, tâche qui semble a priori très simple, mais qui est en fait souvent redoutablement difficile à faire en pratique car elle impose de se confronter à la réalité complexe et mouvante des organisations ; garantir l'exhaustivité et la validité d'une analyse organisationnelle n'est notamment jamais chose facile : il est notamment plus que classique d'oublier certains acteurs clefs et d'en identifier de manière erronée d'autres [29] ; qui plus est, les organisations évoluent souvent rapidement, ce qui fait qu'une cartographie d'acteurs doit toujours être réactualisée en permanence si elle veut rester en phase avec la réalité ;
- *de mettre d'accord ces parties prenantes sur une même solution architecturale*, ce qui est une vraie compétence de « facilitation » au sens noble de ce terme car ce travail nécessite un vrai savoir-faire technico-humain : un architecte ne réussira en effet à faire converger des acteurs sur des choix d'architecture que s'il joue de manière cohérente à la fois sur le plan technique, qu'il doit parfaitement maîtriser pour être crédible, et sur le plan humain, pour arriver à obtenir des consensus solides capables de résister à l'épreuve du temps ;

qui sont typiquement les pré-requis indispensables à la réussite de tout travail de convergence d'acteurs dans le contexte d'une démarche d'architecture système.

L'outil de base de la convergence architecturale est l'*atelier d'architecture collaborative* dont nous allons maintenant rapidement esquisser le mode de fonctionnement de façon à mieux appréhender ce qu'est la dimension « humaine » de l'architecture système. Le principe d'un tel atelier est assez simple puisqu'il consiste « simplement » à mettre dans la même salle « toutes » les parties prenantes que l'on doit faire converger sur une solution architecturale pour que celle-ci soit raisonnablement pérenne [30] et à leur soumettre une première version de l'architecture cible visée. Le travail de fond d'un atelier d'architecture collaborative consiste alors à débattre et à faire évoluer collectivement l'architecture proposée de manière à ce que l'architecture finale qui résulte de ce processus [31] soit réellement le fruit du travail de tous, ce qui fait qu'elle sera naturellement partagée par tous les acteurs présents qui en deviennent donc pleinement parties prenantes. On notera à ce propos que l'architecte système a un rôle

[29] En analysant typiquement mal le rôle d'une personne dans une organisation.

[30] Cela suppose que ces parties prenantes aient été identifiées exhaustivement et correctement, ce qui est en soi un exercice de style difficile car il comporte de multiples chausses trappes (cf. plus haut). Par ailleurs, on conçoit également aisément qu'il est difficile de mener à bien un travail de convergence avec 150 personnes ! La mise en œuvre effective d'un atelier de convergence suppose donc aussi qu'on ait au préalable effectué un réel travail d'architecture organisationnelle en ayant suffisamment abstrait le « terrain » d'une architecture donnée pour ne faire apparaître qu'un nombre limité (typiquement de l'ordre de la quinzaine au maximum) d'acteurs de premier niveau, réellement représentatifs de l'ensemble du territoire organisationnel du système cible, avec qui on fera le travail de convergence souhaité. Ce mode d'analyse est bien sûr récursif et doit être adapté, en respectant une double cohérence technico-organisationnelle, à tous les niveaux systémiques en jeu.

[31] Pour mener à bien un tel travail en pratique, une manière simple de procéder consiste typiquement à rendre visible de tous la modélisation architecturale proposée en imprimant les différents points de vues architecturaux sur de grands posters A0. On peut ensuite facilement débattre et faire évoluer collectivement cette architecture initiale en annotant directement ces diagrammes architecturaux (cf. Figure 9). On notera que c'est l'architecte système qui doit avoir la main dans ce processus de modification « en live » de l'architecture proposée pour garantir en permanence que les choix proposés ont d'une part l'assentiment de tous les participants et d'autre part ne conduisent pas à des sous-optimalités au niveau système.

clef dans ce processus puisqu'il doit fondamentalement garantir que l'architecture sur lequel tous les acteurs convergent reste bien une réponse satisfaisante aux besoins systèmes.

On notera que ce mode opératoire suppose un pré-requis clef, i.e. celui que tous les acteurs partagent le même langage de représentation systémique, autrement dit que la sémantique du mode de description des systèmes qu'on utilisera soit commune à tous, ce qui n'est en général pas gagné d'avance. Il est donc plus que recommandé de commencer un atelier d'architecture collaborative en repartageant avec l'ensemble des participants la sémantique de chacun des éléments de représentation systémique que l'on utilisera [32] . Une fois ces bases bien établies, on peut attaquer le travail de convergence sur une architecture partagée en commençant par analyser collectivement l'architecture initiale proposée. On constate alors souvent en pratique que les premiers problèmes sur lequel on bute sont à nouveau des problèmes syntaxiques car le vocabulaire qui est utilisé pour décrire les éléments constitutifs d'une architecture n'est pas nécessairement partagé ! Pour résoudre cet autre problème, il peut donc être utile de partager avec tous les participants un glossaire des termes techniques utilisés de manière à être sûr que ces termes ont bien la même signification pour chacun des acteurs présents.

Une fois ces derniers obstacles levés, on peut enfin considérer qu'on dispose une base solide pour attaquer le travail technique à proprement parler : celui-ci va donc consister à discuter l'architecture proposée et à la modifier collectivement (cf. l'exemple de la Figure 9), ce qui fait que l'architecture finale qui résultera du débat sera partagée par tous à la fin de l'atelier. On notera qu'il est nécessaire de prévoir un mécanisme d'arbitrage auquel on fera remonter tous les points qui ne font pas convergence lors de l'atelier d'architecture collaborative, s'il y en a, pour prise de décision (auquel cas un nouveau rebouclage avec les acteurs concernés est nécessaire pour motiver les raisons des choix finaux retenus et revalider l'architecture finale avec eux). Comme on le voit, le processus technique d'architecture système s'intègre ici au sein d'un processus d'ingénierie « humaine » sans lequel il ne permettrait pas d'aboutir à des architectures consensuelles et donc « humainement » robustes (ce qui est très souvent une condition sine quo non pour qu'elles soient techniquement robustes).

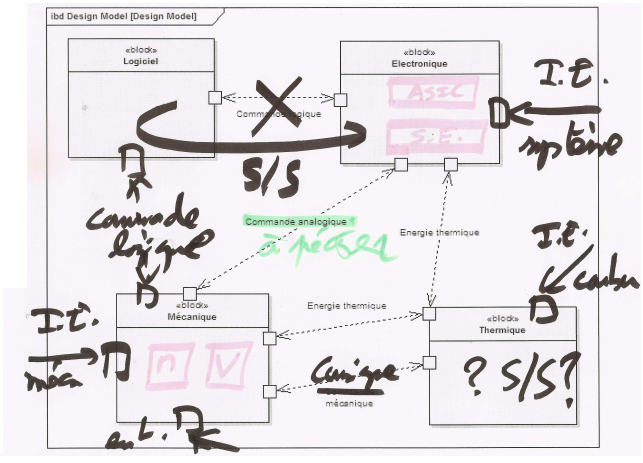

Figure 9 – Un exemple de résultat [33] d'un travail de convergence architectural

[32] En partageant typiquement une légende de tous les symboles utilisés.

[33] Il s'agit ici d'une représentation SysML de très haut niveau d'un système de contrôle / commande édité sur un poster A0. Les annotations au marker sont notamment le fruit typique du travail d'architecture collaborative. On notera notamment que le résultat stricto sensu du travail d'architecture collaborative n'est pas tant le diagramme ainsi obtenu que le fait que les participants à l'atelier ont convergé sur ce diagramme !

5. Quelques éléments de conclusion

Nous avons essayé dans ce très court article de présenter quelques unes des lignes de force les plus importantes de l'architecture système. Notre présentation ne prétend cependant nullement à l'exhaustivité et de fait nous avons été obligé de faire l'impasse sur de très nombreux autres sujets – comme l'ingénierie des exigences, la gestion des interfaces, le dimensionnement des systèmes, le design robuste, la vérification, la validation et la qualification opérationnelle des systèmes, le management des équipes systèmes, la gouvernance des systèmes, la conception de familles de systèmes, etc. – pourtant tout aussi cruciaux pour mettre efficacement en œuvre une démarche d'architecture système. Le lecteur désireux d'en savoir plus sur ces différentes autres problématiques pourra notamment utilement consulter à cet effet les ouvrages [2], [4], [5], [6], [7], [8], [12], [13], [15], [16], [17], [18], [19], [23], [24], [25] ou [26].

Pour conclure notre propos, soulignons enfin à nouveau que la maîtrise de ces savoirs et de ces savoir-faire est désormais clairement un enjeu majeur de compétitivité dans un monde où l'intégration est devenu le paradigme industriel de référence, ce qui fait que la complexité des systèmes industriels ne va faire que continuer à augmenter. On ne peut donc malheureusement que regretter – comme nous l'avons déjà souligné plus haut – qu'ils ne soient pas encore assez compris, diffusés et utilisés en France tant au niveau de la formation initiale de nos ingénieurs que dans nos entreprises. Espérons donc que notre modeste contribution permettra d'aider à faire évoluer cette situation dans la bonne direction !

6. Références

[1] BLANCHARD B.S., FABRYCKY W.J., *Systems engineering and analysis*, Prentice Hall, 1998.

[2] CASEAU Y., *Performance du système d'information – Analyse de la valeur, organisation et management, Neuf scènes de la vie quotidienne d'un DSI*, Dunod, 2007.

[3] COUSOT P., *Interprétation abstraite*, TSI, Vol. 19, n°1, p. 1-9, Hermès, 2000.

[4] D'HERBEMONT O., *La stratégie du projet latéral – Comment réussir le changement quand les forces politiques et sociales s'y opposent ?*, Dunod, 1998.

[5] GALBRAITH J.R., *Designing organizations – An executive guide to strategy, structure and process*, Jossey Bass, 2002.

[6] GRADY J.O., *System Validation and Verification*, CRC Press, 1998.

[7] GRADY J.O., *System Requirements Analysis*, Academic Press, 2006.

[8] GRADY J.O., *System Verification – Proving the Design Solution Satisfies the Requirements*, Academic Press, 2007.

[9] FRIEDENTHAL S., MOORE A. STEINER R., *A practical guide to SysML – The Systems Modeling Language*, Elsevier, 2008.

[10] KAHN G., *The semantics of a simple language for parallel programming*, Proc. of the IFIP Congress 74, p. 471-475, 1974.

[11] KROB D., *Modelling of Complex Software Systems: a Reasoned Overview*, [in FORTE'2006, E. Najm, J.-F. Pradat-Peyre, V. Viguié Donzeau-Gouge, Eds.], Lect. Notes in Comp. Sci., **4229**, 1-22, Springer, 2006.

[12] KOSSIAKOFF A., SWEET W.N., *Systems engineering – Principles and practice*, Wiley, 2003.

[13] LAMPORT L., *Specifying systems – The TLA+ Language and Tools for Hardware and Software Engineers*, Addison Wesley, 2003.

[14] LE CORBUSIER, *Vers une architecture*, Champs Flammarion, 1995 (ré-édition de l'édition originale de 1923).

[15] LIKER J., *Le modèle Toyota*, Village Mondial, 2006.

[16] MAIER M.W., RECHTIN E., *The art of systems architecturing*, CRC Press, 2002.

[17] MARWEDEL P., *Embedded System Design*, Kluwer, 2003.

[18] MEINADIER J.P., *Ingénierie et intégration de systèmes*, Hermès, 1998.

[19] MEINADIER J.P., *Le métier d'intégration de systèmes*, Hermès-Lavoisier, 2002.

[20] PENALVA J.M., *Sagace : la modélisation des systèmes dont la maîtrise est complexe*, Second international conference on Integrated Logistics and Concurrent Engineering, ILCE'94, p. 261-270, 1994.

[21] PRINTZ J., MESDON B., *Ecosystème des projets informatiques – Agilité et discipline*, Hermès-Lavoisier, 2006.

[22] RAMO S., ST CLAIR R.K., *The Systems Approach – Fresh Solutions to Complex Problems through Combining Science and Practical Common Sense*, TRW Inc., 1998.

[23] SAGE A.P., ARMSTRONG J.E. JR., *Introduction to systems engineering*, John Wiley, 2000.

[24] SIMPSON T.W., SIDDIQUE Z., JIAO J.R., EDS. *Product platform and product design*, Springer, 2006.

[25] SOMMERVILLE I., SAWYER P., *Requirements engineering*, John Wiley, 1977.

[26] TURNER W.C., MIZE J.H., CASE K.E., NAZEMETH J.W., *Introduction to industrial and systems engineering*, Prentice Hall, 1993.

[27] WEILKIENS T., *Systems Engineering with SysML / UML – Modeling, Analysis, Design*, Elservier, 2007.